Auton Agent Multi-Agent Syst (2012) 24:217–220
DOI 10.1007/s10458-011-9169-2

Agents for games and simulations

Frank Dignum

Published online: 23 March 2011

Serious computer games have become increasingly popular; they also require more elaborate and natural behavior on the part of Non-Playing Characters. The more elaborate the interactions among characters are during a game, the more difficult it is to design these characters without the use of specialized tools geared towards implementing intelligent agents in a modular way. This thus seems to be an excellent area for the application of intelligent agent technology, which for the past two decades has been developed based on design concepts such as Goals, Intentions, Plans and Beliefs. A first attempt at connecting game engines with these types of agents has been made with Gamebots [1]. Gamebots provides an infrastructure that allows the interfacing of any agent platform to the computer game Unreal Tournament. Gamebots manages the provision of relevant information regarding the game state, while delivering commands for actions from the agents to Unreal. More recently, this package was used as the basis for more extensive middleware called Pogamut [2].

Although the aforementioned middleware does allow the interfacing of agents to the game engine, that in itself does not guarantee proper *behavior* of the agents in the game. In the workshop series on Agents for Games and Simulations [3,4], started in 2009, issues regarding the connection of agent technology to game engines has been discussed. Most of these issues derive from the fact that game engines are typically designed to be in total control of the game's progress. On the other hand one, of the major attributes of agents developed on MAS platforms is that they are autonomous (to some extent) and interact asynchronously. We want to exploit the benefits of having agents deciding intelligently and autonomously about their next actions, while not losing control of the game. This balancing act leads to three broad categories of issues.

The first category is that of technical issues; an important issue in this category is that of coping with real-time environments. Unfortunately, agent technology has hardly bothered with real-time issues up until now. A major exception is the use of agent technology in robotics, where one obviously also has to deal with real-time environments. Maybe this is one of the reasons that, in robotics, people do not use standard (BDI) agent platforms as basis for

F. Dignum (✉)
Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands
e-mail: dignum@cs.uu.nl

their work, but rather start from specialized robotics platforms and extend these with agent concepts.

Real-time behavior is of paramount importance for games. This becomes particularly clear when dealing with rationality. Agents do not always have time to reason about all aspects of a problem until they find the best option; in real-time systems, agents need to have bounded rationality. This aspect is dealt with extensively in the paper of Rosenfeld and Kraus that appears in this special issue.

Real-time reactive behavior can also become a problem when a massive amount of information about the game state is given to the agent every few milliseconds (as happens in most video game engines connected to agent systems). Many agent platforms assume that an agent will monitor events from the environment and update its belief base whenever necessary. However, when so many events, containing so much information, come in, they flood the agent's perception module. This might lead to paralysis of the agent, as it is continuously dealing with incoming events and has no time for other tasks. It is clear that some kind of filtering should take place. The difficulty is, of course, to filter out all irrelevant information and events, while retaining those that are relevant. What is relevant will differ per agent, and also per situation. For example, the fact that there is an accident on the road near a person's house may not be relevant for that person if he happens to be at work; it *is* important if the accident involved the person's partner. It is also relevant for the medical personnel at the hospital nearby, but only if the accident involved casualties.

Another real-time issue is that of durative actions. In most agent platforms, actions are supposed to be executed instantaneously. Thus an agent can wait for an action to be done before it does anything else. However, if an agent is moving in a virtual environment, an action might take time, and an agent should be doing other things in the meantime. A simple solution is to create an action that starts the durative action, then wait for a special event that is created when the durative action finishes. Although it is possible to use this construction, it is up to the agent designer to keep track of all administration surrounding the durative actions, and have the necessary procedures to react appropriately when they fail.

The last important real-time issue we mention here is the balance between reactive and pro-active behavior. Sometimes an agent is supposed to react instantly to an event (e.g., when he sees an accident happen in the street). In these cases, he should abandon his current plan and form a new plan to handle the situation. In other cases (e.g., when a car breaks down in front of him) he might just change his current plan and take another route to avoid the traffic. At still other times (e.g., when he gets a message that there is a sale starting at his favorite clothing store) he will react to the event only after he finishes his current goals (to work until five o'clock). All of these reactions require a mechanism to balance pro-active and reactive behavior of the agent in a natural way.

Although the above real-time issues are not dealt with explicitly in the papers in this special issue, they are part of the reason why Silverman et al. and Lim et al. created their own agent architectures, instead of using an existing platform.

A second category of issues is concerned with the fact that the deliberation of agents usually takes place at a symbolic level (in order to reason at the strategic level), while most information in the game engine is processed at a geometric level (for reasons of efficient rendering). Thus for the game engine, an object might for example be a rectangle that can rotate along one of its edges, while an agent would like to deliberate about this same object as being a "door". Some translation should therefore be provided between the game engine and the agent platform. This also becomes very important when affective interpretations should be attached to events happening in the world. As an example, someone dropping a glass might get an interpretation of being astonished. In the papers of both Lim et al., and Sollenberger and Singh, the affective components of the agents play an important role. The issue of both interpreting and generating emotions in terms of physical elements becomes important.

The last category of issues regards the design of games that have agents. In [5] it is argued that the current practice in the game industry, where the AI parts of the game are added to a game at the end, leads to a poor and even negative result of the AI for game play. The main reason for this outcome is that the potential of AI techniques consisting of adaptation, learning, and variable behavior cannot be used because the behavior of characters is already too constrained. In order to make full use of AI and agent technology, one should already take the possibilities of these techniques into account at early stages of game design. One of the biggest fears of game developers is that the use of agent technology, with agents behaving autonomously, will lead to games that get out of control. That is, agents will behave in unexpected ways and obstruct the game flow, rather than enhance it. In the approach of Silverman et al., the design of the game is scenario-based, and thus in some sense very agent-centric. Interactions among agents are modeled using rules that indicate what kind of combinations of actions can appear. Thus one can predict quite well what possible situations can arise; the exact environment in which agents operate can be added later. This approach works when using your own platform, and if interactions are limited. When many agents interact, and they have many choices to react to one another, one can easily lose oversight of the interaction, which might result in strange behavior when the game is played. To prevent this from happening, we need a way to specify the desired game flow at a high level, such that it can be used to constrain the behavior of agents without limiting it unnecessarily. Recent developments in the use of agent organizations [6] seem to offer a basis for such solutions, but should be adapted further to game design methodologies.

Besides the above categories of issues that deal with the actual interfacing of agents to game engines, one should also look at the appropriateness of the traditional BDI model for agents in games. The BDI model is, of course, a limited model for human behavior. There is thus a need to incorporate additional aspects into agents in order for them to behave more like humans within the game.

The papers collected in this special issue are all related to extensions and new architectures for agents that have to function in a virtual world environment. The paper of Rosenfeld and Kraus argues that agents that have to interact in a natural way with humans should have bounded rationality. The paper presents experiments with some optimization problems and with negotiation situations that suggest that even though optimal strategies might exist, people do not use them, and also do not expect agents to use them. In the paper it is shown how the use of Adaptive Aspiration Theory can lead to natural behavior. This leads to the conclusion that agents in games should make use of this type of theory rather than of optimization techniques if they want to appear life-like.

The paper of Sollenberger and Singh deals with the affective awareness of agents in games. It is recognized that affective awareness is missing from standard agent architectures. Of course, this is an important issue when an agent has to interact with humans and has to exhibit natural behavior. Rather than presenting a new extended agent architecture that includes affective awareness, the authors present a framework that contains a separate affective awareness module. This module can be connected to more standard architectures and thus influence the behavior of the agent. The solution ensures that designers can concentrate on the different aspects in different modules, and also possibly reuse their designs in other agents.

The paper of Lim, Dias, Aylett and Paiva concentrates on the emotional aspect of agents in virtual worlds. The architecture that is presented builds on the well-known FAtiMa architecture, but combines this with the PSI model for urges. Agents take into account both perception, as well as internal aspects such as motivation, emotion, memory and learning, to direct their behavior. The resulting architecture is tested in a game to teach children to cope with refugee integration problems. Again it is shown that in order for agents to interact naturally with humans in gaming environments, we need a richer and more biologically-driven model than the traditional BDI model.

In the final paper of this special issue, Silverman et al. describe their experiences of using many sociological and psychological models to design different aspects of agents for serious computer games. The paper also has a software engineering aspect in the sense that it advocates the use of Model Driven Architectures for the combination of many modules. Using such a principled approach gives designers some grip on the complexity of combining many (independent) modules, each of which also evolves when gaining experience. Using MDA makes it possible to combine software using meta-models. If a new module is added, one needs to describe a meta-model of the concepts used in that module, and check how the meta-model can be integrated with the meta-model of the other modules.

The articles in this special issue constitute a good introduction for people who want to work in this exciting field of agents for computer games. The field offers many new challenges for agent theory, architectures and platforms. However, it also offers many new opportunities to exploit the existing research of agent technology!

References

1. Adobbati, R., Marshall, A. N., Scholer, A., Tejada, S., Kaminka, G. A., Schaffer, S., & Sollitto, C. (2001). Gamebots: A 3D virtual world test-bed for multi-agent research. In *Proceedings of the second international workshop on Infrastructure for agents*, MAS, and Scalable MAS, Montreal, Canada.
2. Gemrot, J., Kadlec, R., Bída, M., Burkert, O., Píbil, R., Havlíček, J., Zemčák, L., Šimlovič, J., Vansa, R., Štolba, M., Plch, T., & Brom, C. (2010). Pogamut 3 can assist developers in building AI (not only) for their videogame agent. In F. Dignum et al. (Eds.), *Agents for games and simulations trends in techniques, concepts and design*. LNAI 5920. Heidelberg: Springer.
3. Dignum, F., Bradshaw, J., Silverman, B., & van Doesburg, W. (Eds.). (2010). *Agents for games and simulations: Trends in techniques, concepts and design*. LNAI 5920. Heidelberg: Springer.
4. Dignum, F. (2011). *Agents for Games and Simulations II: Trends in techniques, concepts and design*. LNAI 6525. Heidelberg: Springer.
5. Russell, A. (2008). Ecological balance in AI design. In S. Rabin (Ed.), *AI game programming wisdom 4* (pp. 49–57). Brookline: Charles River Media.
6. Westra, J., van Hasselt, H., Dignum, V., & Dignum, F. (2008). On-line adapting games using agent organizations. In *Proceedings IEEE symposium on Computational intelligence and games*, Perth, Australia.